(065) 8910

The Ottawa T.I.99/4A Users' Group

NEWSLETTER

VOLUME 8 NUMBER 8.........October 1989

DON'T FORGET THE MEETING -- October 3, 1989

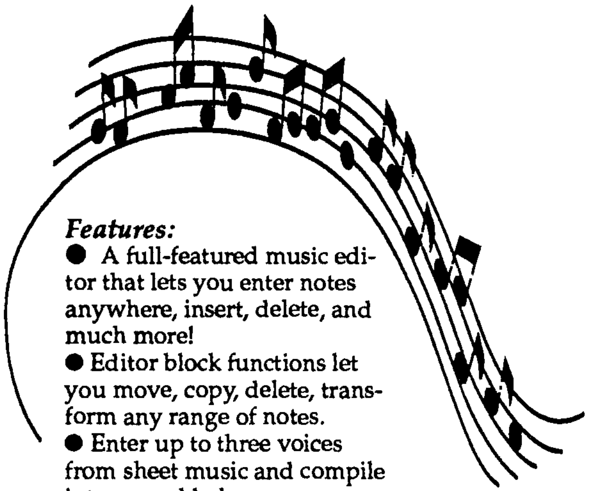
AND -- Remember to return your exchange newsletters!

P.O. BOX 2144, STATION D, OTTAWA
*** ONTARIO, CANADA K1P 5W3 ***

COMING EVENTS

October Meeting:	October 3, 1989 7:30 p.m.	Merivale High School
TI-BASE Workshop:	October 17, 1989 7:30 p.m.	Bill Sponchia's home Contact Bill Sponchia or Tom Bentley for details.
November Meeting:	November 7, 1989 7:30 p.m.	Merivale High School
5th Annual TI-FEST	April 28, 1990	Merivale High School Contact Ruth O'Neill for details or to volunteer your help.
Newsletter Deadline:	October 20, 1989	(or any time before that!)

Music Pro



Music Pro is an amazing MUSIC PROcessor for the TI-99/4A that makes creating music on the 99/4A as simple as typing in sheet music! By David Caron and Lucie Dorais of the Ottawa TI-99/4A Users Group, *Music Pro* allows you to type music directly on a staff - one voice at a time. While editing you can delete, insert, move, copy and even transform and transpose single or whole blocks of notes at a time. Enter up to 400 notes for a single voice. Later, after entering all the voices you can compile them into an assembly language file. *Music Pro* automatically takes care of time differences between the voices - you just concentrate on entering the music and *Music Pro* takes care of the rest! Compiled music can be run from the program or used by itself for games or music programs. Even print out sheet music on an Epson or compatible printer! Requires 32K, Extended BASIC, Disk. *For the TI-99/4A only.*

Features:
- A full-featured music editor that lets you enter notes anywhere, insert, delete, and much more!
- Editor block functions let you move, copy, delete, transform any range of notes.
- Enter up to three voices from sheet music and compile into assembly language - no need to make all notes the same duration between voices! Great for musicians!
- Print out sheet music on Epson or compatible printer.
- Transpose octaves.
- Easily save/load music
- Compiled music can be run independent of program in Extended BASIC!

Suggested Retail:

$17.95

U.S./Canada - Please add $1.00 S&H
Other - Please add $4.00 Airmail

Asgard Software
P.O. Box 10306
Rockville, MD 20850
(703)255-3085

The President's Two Cents' Worth

by Jane Laflamme

Phew! Just got back from the International TI Expo in Washington which was sandwiched between our monthly meeting and the executive meeting! We survived - barely.

It was a great trip, although it took much longer to drive down than we expected -- 15 hours of driving, gassing up, and eating.* The fair was smaller than we expected, but we thoroughly enjoyed it. There were several new releases announced and several hints of those to be released at the Chicago fair. As usual, I managed to miss most of the fair itself by socializing with old and new friends. (I did manage to come back with some updates from Barry Boone and new public domain software he released for the 9640.) I finally met Laura Burns, partner in Burns and Koloen Communications who publish MICROpendium; (I always enjoy meeting other women in this male-dominated field and have been a big fan of MICROpendium!). We also met Barry Boone, author of the Archiver... and of course chatted and/or had coffee with many of the others we have met over the years of attending fairs. There was a good representation from Canada at the fair, including Steve Mickelson and his family, Gary Bowser, Bob Boone, and our delegation of Ottawa members.

A few of us went on our own tour of Washington on Friday via Metro. It only took 8 of us half an hour to master the machines that dispense tickets -- quite an achievement (tourists!). BUT we did manage to arrive exactly where we wanted to go (and even made it back to the hotel -- well, with a small detour). Sunday we were accompanied by Chris Bobbitt of Asgard Software and we showed him our expertise in acquiring tickets (he was not as impressed as we thought he should be). He was definitely a wealth of information. He also managed to acquire free tickets for two IMAX movies at the Air And Space Museum. Thanks, Chris. In brief, on the two days of touring we managed to see a little of the Smithsonian Institute, Capitol Building, Washington Monument, Lincoln Memorial, Vietnam Memorial Wall, White House, Arlington Cemetery parking lot, the top of the Watergate Hotel, various exhibits, statues, houses, gardens, AND the Ruby Slippers from the Wizard of Oz. (Not necessarily in order of importance....) We also walked at least a hundred miles. I may be a little flip in the telling, but this trip was by far the best I have had in the last 5 years of attending fairs. Wish I had had more time, but I hope to return very soon and see things in a little more depth.

And on to business -- the next meeting will feature, as is usual in October, the election of officers who will lead the group into 1990's. It is still not too late to add your name to the list of volunteers, either as an executive or a member of a committee and I remind you that the position of Newsletter Editor is still vacant. We need your help! Contact Bill Sponchia whose telephone number is on the back page, or in the case of Newsletter Editor, Ruth O'Neill. Ruth is willing to help you for a long as you wish, or even will just chat with you if you are at all considering it but are a little hesitant. Time permitting, there will be a demo of Asgard's Legends II by Ruth O'Neill, and Charles Earl will tell us about Gary Bowser's new package for the Horizon RAM disk, called RAMBO. It sounds quite exciting. We will also resume the raffle which I completely forgot last month!

Finally, I would like to thank all those who have helped over the year of my presidency and wish the new members of the executive a good year. You will enjoy the involvement and find a different world of the T.I. will open to you. In other words, as with most things, you will receive more than you give. Thanks everyone.

'Til next ... er ... time...

* Editor's Note: The return trip was considerably shorter -- none of us got lost on the way back. <grin>

Browsing the Library
with Dave Morrison

Although nobody has mentioned the absence of a Library Column in the last edition of our Newsletter, I thought that I had better put in an appearance this month even though there is little to write about. Those of you who attended our first meeting of the season are aware that the Disk of the Month was supplied by our own Lucie Dorais. The disk contained Lucie's Extended Basic tutorials and programmes that she had previously presented in our monthly Newsletter during 1987 and 1988.

My thanks to the Lima, Ohio User Group who very kindly sent a copy of XHi v3.2 and X80 (high resolution graphic and 80 column text support from regular TI Basic, for Geneve and AVPC systems) and FUNNELWEB v4.13 update, 80 column editor update, and enhanced 80 column Quick Directory.

Thank you also, to the Central Westchester 99'ers Club for their quarterly newsletter on disk, the contents of which I shall be discussing in a future article.

At a recent Executive Meeting, it was decided to reduce the "copying fee" for Library disks to $2.00 for SSSD/DSSD with an additional $1.00 charge for the second disk of a SSSD set. I must point out, however, that 1st Class postage to Canadian, United States and foreign destinations has been increased and sufficient funds should be included when ordering by mail. A recent example was a postal charge of Cdn$3.00 for a package of four disks sent via First Class mail (whatever that means) to a destination in the U.S.

Don't forget that Tuesday, October 3rd is ELECTION NIGHT! Turn out and vote for your favorite candidate - even if it is yourself!

ANNOUNCING THE FIRST WORKSHOP OF THE NEW YEAR
by Bill Sponchia

Starting on Tuesday, October 17th, there will be a workshop on how to use the program TI-BASE. This workshop will run for 3 or 4 sessions (or more if needed) with one session per month. The present plan is to hold the sessions on the third Tuesday of each month. The workshop will cover the setting up of a database, the manipulation of data and the generation of reports.

Instructor (guru): Tom Bentley   Assistant: Bill Sponchia

Information for first session:

Date	- October 17, 1989 Tuesday
Time	- 7:30 pm
Place	- 1051 Harkness Ave, Ottawa (Bill's home)
Cost	- donation towards cost of coffee and donuts

Everyone is welcome to attend, but please inform either Tom or Bill if you intend to attend the first session on October 17th. This will allow us to make alternative plans if the demand is too high to be accommodated at Bill's. On that note, please be advised that should you come without calling and the place is too crowded, you will be the one who is asked to leave.

For further information or to advise that you will be attending, please contact either Tom at 826-3306 or Bill at 523-0878.

If you would like to have another Workshop (or whatever you wish to call it), or wish to present one yourself, please let your Executive know. Either that or contact Bill Sponchia and we will see what can be done. Remember - it's up to you.

EDITOR'S NOTES from **Ruth O'Neill** -- Rather than bore you with complaints about the lack of a new editor (needed for next issue, mind you), this month I have two relevant editorial-type items: an announcement from Asgard Software regarding a recent incidence of piracy, and a letter from CaDD Electronics. I hope you will find them to be of interest, as I did.

ANNOUNCEMENT CONCERNING HARDMASTER

We at Asgard Software recently learned that a program we were preparing for release, HardMaster, has already been "released".

Approximately 2 weeks ago, a program purporting to be the "Myarc HFDC Sector Editor" (HFDCSE) appeared on GEnie and on several bulletin boards, along with extensive documentation. This program is, in fact, a program known as HardMaster and it is by Colin Christensen of Australia.

Someone obtained a copy of this program by unknown means, as well as the documentation I myself prepared from information provided by the author, and removed all references in both to the author, Asgard Software, and the real name of the program. The culprit substituted in Myarc's name for that of the distributor and the author and the name of the program.

The perpetrator then placed the program on bulletin board services, and eventually it was placed (either unintentionally or otherwise) on GEnie where it received national distribution.

Many HFDC users have unwittingly downloaded this "gift from Myarc", and spread it to other bulletin boards. No doubt now the program is also available in some user group libraries. This is, of course, illegal (though innocently done). All honest users, user groups and BBS owners are required by law to immediately remove this program from their collections and eliminate all copies of it. If you like the program, and would like a legitimate copy, you can send a check for $14.95 to Asgard Software (P.O. Box 10306, Rockville, MD 20850) and we'll be happy to send you a legitimate copy of the program with a nicely printed manual.

I believe that most users condemn piracy. I also know that some condemn it but wink at it when it benefits them. This particularly insidious form of piracy benefits NO ONE. This type of action strikes at the very core of the right to ownership - the right to be acknowledged as the author of a program. If authors cannot be assured that the programs they write will contain notice of their authorship, regardless of how the program is distributed, THEY WILL NOT WRITE THEM. Period. No author likes to see his/her program pirated, but if it has to happen they at least like people to know they wrote it. This doesn't even give the author that much.

This form of software piracy does not benefit the honest user or even the sometimes pirate - it only benefits those that wish to bury the TI community once and for all. Therefore this action, and all actions like it, should be CONDEMNED BY ALL SOFTWARE USERS AND MANUFACTURERS.

We'd like to thank the SYSOPs of the three major networks, Barry Boone of GEnie's TI Roundtable, Jeff Guide of Delphi's TI Information Network and Jim Horn of Compuserve's TI Forum, as well as the several BBS owners we've contacted, for their prompt action in helping to stem the distribution of the pirated version. We'd particularly like to thank Barry Boone for pointing it out to us in the first place.

Finally, I'd like to note that we have retained a lawyer for this matter, and criminal as well as civil legal action will be taken against the individual or individuals responsible for this act (or their legal guardians). We also expect that all users who unknowingly helped transmit this program will provide us with information about where they obtained it. We will of course not prosecute any user who provides information of this type. Again, the fault for this incident lies with the person or persons who stole and modified the program and documentation in the first place, and not those individuals who took the accreditation of the program to Myarc at face value.

Thank you.

Chris Bobbitt, Asgard Software

The following is Mr. Van Coppenolle's letter to the editor. We would like to thank him for taking the time to provide these clarifications and additional information about the Gramulator.

My name is Mark Van Coppenolle, owner of CaDD ELECTRONICS. I am writing to you in response to an article which appeared in your Volume 8 Number 6 June 1989 newsletter. The title of the article was "The Gramulator: a report". The article was written by Mr. Lawrence Wade.

Although the article was well written and in general gave the Gramulator a very favorable review, there were a few points left open. My wish is to try and answer these points. Hopefully my answers may clear up some confusion and not create it.

I will approach each point in the order Mr. Wade listed them.

1) Mr. Wade was wondering why there is no description in the instruction manual on how to load a cartridge saved to disk under Extended Basic.

When TI developed GROM they also developed Graphics Programming Language (GPL) at the same time. Since GROM can not be directly accessed by the CPU a GPL interpreter was written. This interpreter takes advantage of the features in GROM chips, specifically Auto-incrementing addressing. To run GPL code in normal CPU RAM would require approximately a 20% rewrite of the GPL interpreter. The lack of information on the GPL interpreter would make this, at the very least, difficult. If this rewrite was successful, the "new" interpreter would then have to be linked to each saved cartridge. This creates three immediate problems. One, the size of the cartridge is limited to 32k minus the size of the "new" interpreter. Two, any cartridge which accesses memory expansion (i.e. TI Writer) would not work. Three, in general this would be more work than the public, in my opinion, is willing to do to get a saved cartridge to run. It is simply easier for the user to have a piece of hardware that performs as if it were a cartridge.
We have never stated that the user could run a cartridge without the Gramulator and to CaDD's knowledge no other GRAM emulating device makes this claim.

2) Mr. Wade states that there is no description of the "Loader" or the "RAM1/RAM0 switches. In addition he has questions about the "Bank Switching" and the "Write Protect" switches.

I refer Mr. Wade to page eighteen in the instruction manual. This page is intitled "Explanation of Switches". As the title implies, this page explains all the front panel switches and their effects. The page number remains constant for all revisions of the manual. It is also listed in the Table of Contents at the front of the manual.

3) Mr. Wade complains about typing becoming awkward with the Gramulator plugged in.

This, I must concede, is true. It is a minor problem for me, but for others I understand the problem. Any object, that is bigger than a TI cartridge can be a nuisance to some typists. He also made a suggestion to me which I do not understand. Mr. Wade please clarify what you mean by "**ROUND THE EDGES**". The edges of the sheetmetal box are already sanded and painted, and there are no sharp edges which I know of. Can you clarify this? * (Editor's note)

4) Mr. Wade also stated that the battery catches the top of the cartridge port.

It is true that the battery does barely catch on the top of the cartridge port, and the manual does warn you of this. This should not be a problem since the Gramulator does not need to be removed very often. If the user wishes to run a cartridge, he simply plugs the real cartridge into the cartridge connector on the Gramulator. By turning a front panel switch to the "Cartridge" position the real cartridge will run. Also, if a little care is take when you do remove the Gramulator the battery will not catch.

I wish to thank Mr. Wade for his favorable "report". We also encourage other people to make suggestions or complaints, since this is the best way for us to **improve our products.**

* On checking with Mr. Wade, by "round the edges", he was actually referring to possibly blunting the square corners, not any unfinished edges on the device.

ASSEMBLY UTILITY PROGRAMS FOR EXTENDED BASIC
By David Caron

This article marks the beginning of a new series of articles involving utilities. Over the years of assembly programing (with Extended Basic of course), I have built up quite a collection of utility programs which enable Extended BASIC programmers to do nice things like instantly saving and restoring display screens, making screen dumps out to the printer with the characters' 8X8 hexadecimal definitions being included, as well as other "goodies". If you own a copy of Music-Pro, you will notice that all the utilities to come are currently being used by that program. The utilities, once loaded into the low 8K block of memory, can be accessed and used anywhere in an Extended BASIC program with CALL LINK commands.

Below is the assembly code for four routines which involve reading and writing to and from the screen. To use them, simply type the code into an editor like FunnelWriter, assemble the source code into an UNCOMPRESSED object file and load that object file into the Extended Basic environment using: CALL LOAD("DSK*.<filename>") where * is the drive the object file is located on and <filename> is the name of the object file created from the assembler. By the way, an assembler is included with the FunnelWeb system which can run from the Extended BASIC module, so not having the editor assembler module is no excuse for not being able to create an assembly file.

The assembly utilities used in this program, VDPWA, VDPWD etc.. have been explained in articles which can be found in the November 1988 and January 1989 newsletters.

The asterisk "*" in front of text indicates the starting of a comment, SO IT IS NOT NECESSARY TO TYPE THE COMMENTS IN -- the assembler just ignores them anyway. Things like MOVB R2, *R1 are exceptions and are NOT comments.

```
       DEF   RSCRN
       DEF   WSCRN
       DEF   ROLLUP
       DEF   ROLLDN
VDPWA  EQU   >8C02
VDPWD  EQU   >8C00
VMBW   EQU   >2024
VSBW   EQU   >2020
VSBR   EQU   >2028
VMBR   EQU   >202C
NUMASG EQU   >2008
STRASG EQU   >2010
STRREF EQU   >2014
NUMREF EQU   >200C
XMLLNK EQU   >2018
KSCAN  EQU   >201C
NEWREG BSS   32             *My register workspace
STRBUF BYTE  224            *defines size of string
       BSS   224            *reserved memory to create the 256 byte strings.
VDPBUF BSS   28             *reserved memory to temporarily hold one screen row.
NUM224 BYTE  224            *CONSTANT
****************************************************************************
* This code returns control back to Extended Basic after the utility is done.
RETURN LWPI  >83E0          *switches register workspace back to GPL (Interpreter)
*                           * (the program which interprets Extended Basic itself)
       CLR   @>837C         * Indicates no errors on return to Extended Basic
       RT                   *Jump back to Extended Basic

****************************************************************************
* Reads the display screen and dumps it into 3 224-byte strings. Each string
* holds 8 rows (3X8=24) of screen text. Only the Text screen, NOT the graphics
* screen is saved. (8 lines X 28 characters = 244 bytes)
* EX-ACCESS: CALL LINK("RSCRN",<string1>,<string2>,<string3>)

RSCRN  LWPI NEWREG          *switches to my workspace
```

```
       CLR  R3              *R3=0
       CLR  R4              *R4=0
IBLOCK LI   R0,2            *R0=2
       A    R3,R0           *Start saving at row 1, column 3 of screen, R0=VDP add
       AI   R3,256          *next pass starts at 9th line
       LI   R1,STRBUF       *R1= starting place for extended basic string
       MOVB @NUM224,*R1     *SET first byte of STRBUF to 224, (length of string)
       INC  R1              *R1= actual starting place for screen characters
       LI   R2,28           *R2= number of consecutive screen bytes to be copied
IROW   BLWP @VMBR           *Do it!

       AI   R1,28           *R1=Starting place for next piece of string
       AI   R0,32           *R0=VDP address, goes to next line
       C    R0,R3           *Does R0 point to line 9 ? n
       JLT  IROW            *If not then goto IROW

       CLR  R0              *R0=Array element=0; since an array is not being sent
*                           *   to extended basic
       INC  R4              *R4=R4+1  R4=Argument  number (first, second or third)
       MOV  R4,R1           *R1=Argument number, as seen by @STRASG
       LI   R2,STRBUF       *R2=Location of string to be sent to Extended Basic
       BLWP @STRASG         *SEND STRBUF to EXTENDED BASIC

       CI   R3,768          *Does R3 point past 24th line of screen ?
       JLT  IBLOCK          *If no then goto IBLOCK

       B    @RETURN         *goto RETURN
*************************************************************************************
*Opposite of RSCRN, WSCRN reads in three extended basic strings and displays
* them on the screen
* EX-ACCESS: CALL LINK("WSCRN",<string1>,<string2>,<string3>)

WSCRN  LWPI NEWREG

       CLR  R3              *same usage as RSCRN
       CLR  R4              *same usage as RSCRN

OBLOCK CLR  R0              *Array element,set to zero since no arrays being passd
       INC  R4              *Argument number
       MOV  R4,R1           *R1=Argument number  where @STRREF can see it

SKIP36 LI   R2,STRBUF       *R2= start of memory where the EX string should be put
       MOVB @NUM224,*R2     *(224) is the maximum  length of the string.
       BLWP @STRREF         *GET STRING FROM EXTENDED BASIC

       LI   R0,2            * similar to RSCRN
       A    R3,R0
       AI   R3,256
       LI   R1,STRBUF+1
       LI   R2,28
OROW   BLWP @VMBW           *opposite of @VMBR, @VMBW writes to the screen

       AI   R1,28           * similar to RSCRN
       AI   R0,32
       C    R0,R3
       JL   OROW

       CI   R3,768
       JL   OBLOCK

       B    @RETURN
*************************************************************************************
* Reads in one string of size 224 bytes and adds it to the bottom of the screen
* while the rest of the screen is scrolled up 8 lines.
* EX ACCESS: CALL LINK("ROLLUP",<string>)

ROLLUP LWPI NEWREG          *switch to my register workspace
       CLR  R0              *R0=0 since no arrays are being transferred
       LI   R1,1            *R1=1 indicates argument number.
       LI   R2,STRBUF       *R2=memory where EX string is to be put
       MOVB @NUM224,*R2     *first byte in STRBUF contains max length of string
       BLWP @STRREF         *GET STRING FROM EXTENDED BASIC
       LI    R4,STRBUF+1   *R4=starting of line 1 in string

SCRLUP LI    R1,VDPBUF     *R1=start of temporary 28-character buffer
       LI    R0,34         *R0=VDP address (start of second screen line)
       LI    R2,28         *R2=28 (number of bytes to be copied)

ROWUP  BLWP @VMBR          *COPY FROM VDP memory TO VDPBUF
       AI    R0,-32        *R0=start of line above the line copied to VDPBUF
       BLWP @VMBW          *COPY VDPBUF TO above line in VDP memory
       AI    R0,64         *R0=R0+64 (start of line+2 from last line)
       CI    R0,770        *Does R0 point to line after line 24?
       JLT   ROWUP         *If not then goto ROWUP
       AI    R0,-32

       MOV   R4,R1         *R1=starting of line in string (STRBUF)
       BLWP @VMBW          *COPY 1 LINE FROM STRBUF TO 24TH LINE OF SCREEN
       AI    R4,28         *R4= next line in STRBUF
CONST1 CI    R4,STRBUF+224 *Does R4 point past last line in STRBUF?
       JLT   SCRLUP        *If not then goto SCRLUP

       B     @RETURN       *goto RETURN

*The value 244 at CONST1 can be changed to scroll by 8 lines or less.
*The new value can be calculated as VALUE=lines to scrollX28
*EX. 224=8X28

*****************************************************************************
* Same as ROLLUP except the screen is scrolled down while the new EX string is
* added in from the top of the screen.
* EX ACCESS: CALL LINK("ROLLDN",<string>)

ROLLDN LWPI NEWREG         *switches to my registers

       LI    R0,0          *same as ROLLUP
       LI    R1,1
       LI    R2,>E000
       LI    R2,STRBUF
       MOVB @NUM224,*R2
       BLWP @STRREF

CONST2 LI    R4,STRBUF+197 *Start of last line if STRBUF

       LI    R2,28         *R2=number of bytes to be transferred
SCRLDN LI    R1,VDPBUF     *R1=Start of temporary line buffer
       LI    R0,706        *R0=Start of line 23 in screen

ROWDN  BLWP @VMBR          *Copy last line of screen into VDPBUF
       AI    R0,32         *R0=go down one line
       BLWP @VMBW          *Copy VDPBUF to second last line of screen (only pass
       AI    R0,-64        *go up two lines
       CI    R0,0          *Does R0 point before first line in screen?
       JGT   ROWDN         *if not then goto ROWDN

       LI    R0,2          *R0=start of first line
       MOV   R4,R1         *R1=Start of last line in STRBUF
       BLWP @VMBW          *COPY last line in STRBUF to line 1 in screen
       AI    R4,-28        *GO back one line in STRBUF
       CI    R4,STRBUF     *Does R4 point to before first line in STRBUF
       JGT   SCRLDN        *IF not then goto SCRLDN

       B     @RETURN       *goto RETURN
       END

*The value 197 at CONST2 can be changed to scroll by 8 lines or less.
*The new value can be calculated as VALUE=(lines to scrollX28)-27
* EX. 197=(8X28)-27
```

IF you have any question about this utility or the other ones to come, do not hesitate to call me at 837-1397 (Take special notice, the phone number has changed!)

Chainlink -- A Review
by Ruth O'Neill

Chainlink is a delightful new program from JP Software (Formerly Genial Computerware). It was written by Walt Howe and Wayne Stith, with documentation by Walt Howe and is a simulation game, based on the card game Chainlink.

Chainlink is a form of solitaire in which all fifty-two cards are laid out face up, so there is no element of chance beyond the initial deal. The cards are arranged in thirteen columns of four cards, and the object of the game, as with most games of solitaire, is to move each suit of cards up to the top of the table in order, starting with the aces. To do this, a player builds "chains" of cards of the same suit in the various columns of cards. It isn't at all hard to play, but there is a fair amount of skill involved in playing it well.

Obviously, it would be possible to play such a game of solitaire with an ordinary deck of cards, but it is much more fun to use this program. Dealing the cards out is painless -- they deal themselves out before your eyes very quickly. If you're still more impatient to begin the game, you can turn off the visible deal and just have the cards appear before you instantly, ready for play. Personally, I like to watch the cards slide into position, and it takes only a few seconds.

The mechanics of play are surprisingly simple: The columns of cards are labelled with the letters a to m, and moving a card from one column to the other is merely a matter of pressing first one letter, then the other. The card to be moved slides smoothly and quickly into its new position. If you change your mind before you make another move, the oops key (fctn-1) puts the card back for you. If you have a lot of cards to move, you don't even have to go to all the trouble of pressing two keys -- pressing "R" will repeat your move as many times as you like (or until it is no longer possible, of course). Overall, it is a carefully planned, effective, and user-friendly interface. The only thing you can't do that you can it a real game is cheat, but since everybody knows that only the most despicable of card players cheats at solitaire, there's no problem there.

The excellent graphics and pleasant sounds that provide feedback for the player add to the game's enjoyment. It may not be a great game strategy, but I tend to prepare as many cards in order as I can before I press the Enter key and move them to the top of the screen. All those enthusiastic bleeps and bloops and flying cards are immensely satisfying after a lot of thought and hard work.

There are other elements to the software package besides just playing the game, though. It is possible to give up on the current game and ask for a redeal at any point. Chainlink will also keep track of your wins during the session for you, so you can see your progress in mastering the game. You can even save your game, although only in the form of the original deal, not your current position. This has other advantages, though: If you make a crucial error and realize it at some point later in the game, you can start over with a game you know is possible to solve. This also helps if you want to practice a particular game to demonstrate to a friend and show off your skill. The package comes with a large number of saved deals that are guaranteed possible to solve. The only problem I encountered there was minor -- The default to load in these games was "DEAL1", while the first game was actually on the disk as "DEAL01". Not too hard to figure out, but irritating, just the same.

The documentation that comes with Chainlink is excellent. It isn't long, but it doesn't need to be. The instructions are straightforward and clearly presented, and I especially appreciated the tips on game strategy. On the other hand, if you are the type to religiously avoid reading manuals, you still don't have to worry. A summary of the rules is available on screen when you start the game.

All in all, I was tremendously impressed with Chainlink. I've spent many hours playing it (yes, it is definitely addictive) and look forward to many more. The only real problem I've had so far is that it attracts back seat drivers just like the real thing. Oh, well. I've had more than my money's worth from it.

Look for Chainlink from your local dealer (Laflamme & Wrigley is a distributor for JP Software), or you can order it directly from JP Software/ 2390 El Camimo Real, #107/ Palo Alto, CA 94306. The price is $12.00 U.S., plus $1.00 shipping and handling for U.S. and Canadian customers. Canadian customers should always use money orders for U.S. dollars.

A Look At Assembler Language -- String Searching
by R. A. Green

In many programs you find that you have to search down a string looking for a specific character. For example, you've read in a variable length record and want to find the first blank character. You might write the following code.

```
        LI     R1,RECORD        R1->the string
        MOVB   *R1+,R2          Load length of string
        SRL    R2,8             Make length a word
        A      R1,R2            R2->just past end
LOOK    CB     *R1+,@BLANK      Look for 1st blank
        JEQ    FOUND            Jump if we found it
        C      R1,R2            At end of string?
        JL     LOOK             Jump no, keep looking
        JMP    NOBLNK           Jump yes, no blank in string
FOUND   DEC    R1               R1->1st blank
        ....
RECORD  BSS    81               Length, string
```

Seems straightforward enough, but we could do better. Often this type of code can be improved by insuring that you will find what you are looking for. If you are sure you will find something, you don't have to check for the end of the item being searched. The example below shows this. It is also coded using macros (supplied with your favorite macro assembler) to demonstrate again how the use of macros reduces the third cost of a program.

```
        LI     R1,RECORD        R1->the string
        LDB    *R1+,R2          Load length byte
        A      R1,R2            R2->just past end
        MOVB   @BLANK,*R2       Put blank at end
LOOK    IFB    *R1+,NE,@BLANK,LOOK Find a blank
        DEC    R1               R1->1st blank
        IF     R1,EQ,R2,NOBLNK Jump if at our blank
        ....
RECORD  BSS    82               Length, string, 1 blank
```

By adding the extra MOVB to put the blank at the end of the string (and allowing for that blank in the BSS), we have made a two instruction loop rather than a four instruction loop. We have also done away with the label "FOUND". If this is in a time critical part of your program, the extra word of storage used is worth halving the time in the loop.

This demonstrates a general principle (if there are any) that in well-written programs you can usually trade storage cost for time cost or vice versa. Obviously, you trade in favour of storage in little-used parts of the program and in favour of time in the inner loops. Don't be too quick to trade, though, because as we have shown in this series of articles, you can often improve both costs with a little thought. Once you have expended the effort to think about a coding problem and have "learned" how to code it efficiently, or, better still, have encapsulated it in a macro, then all future programs benefit.

Next time we will look at a variation of this same problem which I'll call "counting".

"Write a program that even a fool can use, and only a fool will want to."

HELP WANTED

Wanted: Ottawa TI-99/4a Users Group members to serve on the 1989-1990 executive. Desirable positions are still available. Act now! This offer will not end until you do. Call Bill Sponchia 523-0878. Also urgently required: 1 Newsletter editor, preferably human.

FAST EXTENDED BASIC

LUCIE DORAIS

Shall we dance? Or simple Waltz choreography, to show off the user-defined subs...

The waltz is a three-time rhythm, often played as a simple melody in the Treble Clef, and an accompaniment in the Bass Clef; most often, the chord in the accompaniment is broken into the tonic note played alone, then the two other notes of the chord played together twice, for the characteristic "oom-pah-pah" sound. And all bars follow the same pattern, at least in the easiest arrangements, which is what I have used here...

For the choreography, we have two couples that will whirl and spin around the screen, following the beat. To make the system "universal", i.e. easy to apply to any simple waltz, I have used a SUB BAR that is called once for each bar in the music. The sub plays the notes and moves the two sprite couples. To make it easy to adapt to other songs, I have also done the initialization part as a sub START, called only once.

```
100 ! ** FAIRY WALTZ ** by L. Streabogg (1835-1886)
110 ! progr. by L.Dorais/Ottawa UG/Sept. 1989
120 !
130 CALL CLEAR :: CALL START("LITTLE FAIRY WALTZ",81,49,81,193)
140 LD=147 :: LFS=185 :: LG=196 :: A=220 :: B=247 :: C=262 :: D=294 ::
    FS=370 :: G=392 :: HA=440 :: HAS=466 :: HB=494
150 HC=523 :: HD=587 :: HE=659 :: HFS=740 :: HG=784 :: HHA=880 ::
    R=40000 :: RV=8 :: CV=13
160 GOTO 170 :: A$,TIME :: !@P-
170 FOR TIME=1 TO 2
180 CALL BAR(RV,CV,-RV,-CV,HB,HB,HC,LG,D,B)
190 CALL BAR(0,CV,0,-CV,HD,HD,G,LG,D,B)
200 CALL BAR(-RV,CV,RV,-CV,FS,FS,FS,A,D,C)
210 CALL BAR(-RV,0,RV,0,FS,R,R,A,D,C)
220 CALL BAR(-RV,-CV,RV,CV,HC,HC,HD,LD,D,C)
230 CALL BAR(0,-CV,0,CV,HE,HE,HA,LD,D,C)
240 CALL BAR(RV,-CV,-RV,CV,G,G,G,LG,D,B)
250 CALL BAR(RV,0,-RV,0,G,R,R,LG,D,B)
260 CALL BAR(-RV,CV,RV,-CV,HG,HG,HFS,LG,D,B)
270 CALL BAR(0,CV,0,-CV,HHA,HHA,HG,LG,D,B)
280 CALL BAR(RV,CV,-RV,-CV,HG,HG,HG,LD,C,A)
290 CALL BAR(RV,0,-RV,0,HFS,R,HA,LD,C,A)
300 ON TIME GOTO 310,350
310 CALL BAR(RV,-CV,-RV,CV,HFS,HFS,HE,LD,C,LFS)
320 CALL BAR(0,-CV,0,CV,HA,HA,HAS,LD,C,LFS)
330 CALL BAR(-RV,-CV,RV,CV,HB,HB,HB,LG,D,B)
340 CALL BAR(-RV,0,RV,0,HB,R,R,LG,D,B):: GOTO 400
350 CALL BAR(-RV,-CV,RV,CV,HE,HE,HE,LD,C,LFS)
360 CALL BAR(0,-CV,0,CV,HA,HA,HE,LD,C,LFS)
370 CALL BAR(-RV,-CV,RV,CV,G,G,G,LG,D,B)
380 CALL BAR(RV,0,-RV,0,G,R,R,LG,R,R)
390 CALL PATTERN(#1,132,#2,128):: CALL MOTION(#1,0,0,#2,0,0)
400 NEXT TIME
410 DISPLAY AT(24,11):"AGAIN? Y" :: ACCEPT AT(24,18)SIZE(-1)
    VALIDATE("YN"):A$
420 IF A$="N" THEN END ELSE DISPLAY AT(24,11):"" :: GOTO 170
9999 !@P+
10000 SUB START(A$,R1,C1,R2,C2) :: CALL SCREEN(2) :: FOR X=4 TO 8 ::
      CALL COLOR(X,16,2) :: NEXT X
10010 X=(28-LEN(A$))/2+.5 :: DISPLAY AT(1,X):A$
10020 CALL CHAR(128,"070F0A0B0B0307070F0F1F1F3F3F3F021C1C08FC1CFC1C1C9C88
      C8C8E8E8E818")
10030 CALL CHAR(132,"3838103F383F383839111313171717 18E0F050D0D0C0E0E0F0F0
      F8F8FCFCFC40")
10040 CALL CHAR(136,"03030103030303030307070F0F1F1F1F0280800080808080C0C0
      E0E0F0F0F080")
10050 CALL MAGNIFY(4):: CALL SPRITE(#1,128,8,R1,C1,#2,132,10,R2,C2)
      :: SUBEND
```

```
10060 SUB BAR(R1,C1,R2,C2,T1,T2,T3,B1,B2,B3)
10070 P2=128-4*(P1=128)  :: P1=128-4*(P2=128)
10080 CALL PATTERN(#1,P1,#2,P2):: CALL MOTION(#1,R1,C1,#2,R2,C2)
      :: CALL SOUND(300,T1,5,B1,10)
10090 CALL SOUND(450,T2,5,B2,10,B3,10):: CALL PATTERN(#1,136,#2,136)
      :: CALL SOUND(300,T3,5,B2,10,B3,10):: P1=P2 :: SUBEND
```

SUB START sets text colors to white on black, displays a centered title, and then defines three sprites, two being a mirror of each other, i.e. a couple with the man facing left and another with him facing right, char. 128-131 and 132-135; the third sprite, for characters 136-140, is the spinning couple seen from the back; as it is shown in silhouette (sprites can have only one color), the direction of their spin is irrelevant: both times, you see only the silhouette of the woman's crinoline.

(I am grateful to Jeremy Frank, from Windsor and Ottawa, for his MAKECHAR program, part of his fairware disk distributed by our Library: not only did it make it easy for me to mirror my first sprite and design the third one by slightly modifying the first one, but it also allowed me to check the animation before committing everything to the program. I also used my own SPRITE TESTER (last month) to find the starting positions of my two sprites. End of commercial.)

The starting sprites are called in line 10050 and are made as big as possible by a magnifying factor of 4. One is pale blue (8), the other pink (10); the third and fourth parameter for each sprite is its starting position: dotrow 81, dotcolumns 49 and 193 respectively, passed to the sub as parameters in the CALL.

Lines 140-150 define the variables used for the notes, from low D to high-high A (above the Treble Clef); the rest R is 40000, and finally we define the row velocity and column velocity of our sprites as RV and CV. This abundance of variables is placed before the Pre-Scan, so the latter, in line 160, deals only with those variables not yet defined; all the CALL statements are used mostly in the user-defined subs, so no need to Pre-Scan them, since the subs are outside the pre-scanning range (line 9999).

The score of our waltz becomes a series of CALL BARs, which carry over a list of ten parameters: row and column velocity for each sprite, then the three notes in the Treble Clef (the same note is repeated to allow for half notes and dotted half notes in the score), and finally the three notes for the Bass Clef broken chords. To make the typing job easier, use the REDO function in command mode: type line 180, the press FCTN 8: the line is brought again at the bottom on the screen; type over the new line number, and type over the parameters that are different. Be careful with the "+-" signs in the choreography, which give opposite movements to our couples.

Even if the sub is complicated, with all its parameters, it saves a lot of code! Just imagine lines 10080-90 repeated 20 times! And, because most of the values passed are different from one bar to the next, a GOSUB solution was not practical.

Since the score has a different ending for the two times the melody is played, we use an ON GOTO statement in line 300 to control the flow of the programmed music. Line 390 simply brings our couples to a much needed rest while Tex asks you if you want to dance again.

Now to SUB BAR, lines 10060-10090: our ten parameters are listed in brackets as R and C velocities for each sprite, then the three Treble Clef notes and the three Bass Clef notes. Line 10070 defines the contrasting patterns for each sprite; P2 is set to 132 each time P1 is 128, and to 128 each time P1 is not 128, including the first time the sub is called, when P1 is still zero. Then P1 is defined the same way: when P2 is 132, P1 will be 128, and vice-versa. The relational expressions in brackets return a "0" if they are false, and "-1" if true. This line uses the notion that the variables in a user-defined sub keep their values from one CALL to the next.

Line 10080 changes the patterns of the sprites before playing the first beat of the bar: one note T1 in the melody, one (B1, the chord tonic note) in the accompaniment; the duration of this first beat has been set to 300. Line 10090 plays the second beat, with a slightly longer duration, as in a true Viennese waltz; the second note of the melody bar and the two upper notes of the Bass chord are played. The sprite pattern is then changed to the "backview" sprite of char. 136, then the third and final beat is played: third melody note, and same two notes as previous beat for the accompaniment (B2 and B3). Before leaving the SUB, P1 is set to P2 so that next time the sub is called, we will get a reverse sprite pattern from the previous CALLed BAR.

To show that this pattern could be used with any waltz, I picked another simple score and, using the REDO function (and the ERASE to delete the lines that I didn't need, since this second score is much shorter), I quickly modified the score portion of the program as follows:

```
100 ! ** L'EAU VIVE ** by Guy Beart
110 ! progr. by L.Dorais/Ottawa UG/Sept. 1989
120 !
125 CALL CHARPAT(39,A$) :: CALL CHAR(61,A$)
130 CALL CLEAR :: CALL START("L=EAU VIVE",106,33,106,209)
140 LB=123 :: LC=131 :: LE=165 :: LF=175 :: LG=196 :: C=262 :: D=294
    :: E=330 :: F=349
150 RV=25 :: CV=20
160 GOTO 170 :: A$,TIME :: !@P-
170 FOR TIME=1 TO 2
180 CALL BAR(-RV,0,-RV,0,E,E,C,LC,LE,LG)
190 CALL BAR(0,CV,0,-CV,E,E,C,LC,LE,LG)
200 CALL BAR(RV,0,RV,0,E,E,C,LC,LE,LG)
210 CALL BAR(0,-CV,0,CV,D,D,D,LB,LF,LG)
220 CALL BAR(0,CV,0,-CV,D,E,F,LB,LF,LG)
230 CALL BAR(-RV,0,-RV,0,D,D,E,LB,LF,LG)
240 CALL BAR(0,-CV,0,CV,C,C,C,LC,LE,LG)
300 ON TIME GOTO 310,350
310 CALL BAR(RV,0,RV,0,D,D,D,LB,LF,LG):: GOTO 400
350 CALL BAR(RV,0,RV,0,C,C,C,LC,LE,LG)
390 CALL PATTERN(#1,132,#2,128):: CALL MOTION(#1,0,0,#2,0,0)
400 NEXT TIME
410 ...        |
420 ...        | same as FAIRY WALTZ
9999-10090     |
```

Besides changing the parameters for each CALL BAR, I modified lines 140 and 150 to use only the notes I needed, and the row and column velocities to meet my new choreography requirements; line 125 has been added to display the apostrophe in the title in the same color as the other characters. The rest of the program is exactly the same as the previous waltz, with one exception: since each song has a different feeling, I changed the durations of the notes, from 300-450-300 to 250-350-250, in the three CALL SOUNDs (lines 10080-10090)

If you want to use this template for other waltzes, you could save lines 9999-10090 as a MERGEable file (this is why I used higher line numbers), remembering to adapt the sound durations to what you want. You could then modify it a bit for a four-beat rhythm. How about a rock beat? but make sure to change the costume of our lady!

ABOUT MY ROOTS...

Shame on me: last month, I gave you what I thought was a clever way to find any root of a given number. Before writing the program, I checked both the XB manual and a book I have, called "Mathematics Made Simple", to be sure there was no easy formula, and the averaging process was the only one... Well, two people came to me at the September meeting to tell me that there IS an easy formula: given the number N and the root R, you find the root with the statement "AN=N^(1/R)".... Thank you Philip and Gabriel!

So here is the same program, but with the new statement in line 180 (delete lines 190-210) and minor changes to lines 170 and 220, to get rid of the variable AV altogether:

```
100 ! ** ROOTS ** L.Dorais/Ottawa UG/July 1989 / mod. Sept. 1989
110 !
120 ON WARNING NEXT :: CALL CLEAR :: PR$="PIO"
130 L$=RPT$("}",28):: E$=RPT$(" ",168):: S$=RPT$(" ",8)
140 CALL CHAR(120,"0000000000002050F",121,"1F10202040408080",
    122,"018182C2C4646830",123,"0B0101",125,"FF")
```

```
150 DISPLAY AT(5,9):"x3y"&RPT$("}",10):S$&"{z" :: GOSUB 280
160 ACCEPT AT(6,12)VALIDATE(NUMERIC)BEEP:N :: IF R>2 THEN 180
170 IF R=1 THEN AN=N :: GOTO 220 ELSE AN=SQR(N):: GOTO 220
180 AN=N^(1/R)
220 DISPLAY AT(12,8)BEEP:"=";AN
230 DISPLAY AT(22,1):L$:" [A]nother   [C]hange root   [P]rint     [Q]uit"
240 CALL KEY(0,K,S):: IF S=0 THEN 240 ELSE K=POS("ACPQ",CHR$(K),1)
250 IF K=0 THEN 240 ELSE ON K GOTO 260,260,270,290
260 DISPLAY AT(7,12):E$:E$:E$ :: IF K=2 THEN GOSUB 280 :: GOTO 160 ELSE 160
270 OPEN #1:PR$ :: PRINT #1:S$&"           ________________":S$&" "&STR$(R)&"/" ::
   PRINT #1:S$&"\/ ";N;TAB(26);"=";AN:"" :: CLOSE #1 :: GOTO 240
280 ACCEPT AT(5,10)VALIDATE("123456789")SIZE(-1)BEEP:R :: RETURN
290 END
```

And if you want a quicker program, here is a tinygram:

```
100 INPUT "NUMBER: ":N :: INPUT " ROOT: ":R :: PRINT "ANSWER:";N^(1/R) ::
    PRINT :: GOTO 100
```

A THOUGHT FOR THE SILENT MAJORITY...

Since this is Election month in our Group, I want to share with you some thoughts by Terrie Masters, of the Los Angeles Group, and reprinted in the Hunter Valley (NSW, Australia) Newsletter:

"... well lots of people like lots of things happening, as long as someone else does them. How about burnout, you all?... If the system starts to wear down, it is NOT the fault of the current doers. If you are one of the myriad of takers, any chance of a switch of roles? Try it, you might like it!"

HOTLINE NUMBERS

Note: Please respect normal hours unless you specifically know that someone doesn't mind a call at 3am, or use the BBS to leave a message at 738-0617, 24 hours a day, 7 days a week.

JANE LAFLAMME	PRESIDENT	(H) 837-1719 or (W) 745-2225
AL PALMER	VICE PRESIDENT	594-9216
MARCELLE GIBSON	SECRETARY	233-2384
BILL SPONCHIA	TREASURER	523-0878
MICHAEL TAYLOR	PAST PRESIDENT	831-0143
PETER ARPIN	SYSOP	523-0017
RUTH O'NEILL	NEWSLETTER EDITOR	234-8050
TONY HOPKINS	ADVERTISING	746-4463
DAVE MORRISON	LIBRARY CHAIRMAN	737-4889
JACK McALLISTER	CASSETTE LIBRARY	225-6989
HENRI MONAT	ARCHIVES	824-0941
LUCIE DORAIS	MEMBERSHIPS	232-0393
BOB BOONE	HARDWARE/SOFTWARE	(705) 476-9391
ART GREEN	ASSEMBLY HELP	837-1955
DICK PICHÉ	TECH	521-8667
DAVID CARON	TECH, EXTENDED BASIC, ASSEMBLY HELP	837-1397
CLUB BBS	SET MODEM TO 8N1	738-0617

LAFLAMME & WRIGLEY WHOLESALE

5480 Canotek Road, Unit #16
GLOUCESTER, ONTARIO
K1J 9H6
TEL: (613) 745-2225 Fax: 744-4784

LAFLAMME & WRIGLEY DATA PHONES:
DELPHI "JANELAFLAMME" (no space)
CompuServe "76046,2006"
Ottawa BBS(TEXLINK) (613) 738-0617

Authorized Distributor for the following:

ASGARD Software **GENIAL COMPUTERWARE**

Inscebot, Inc. **MYARC, Inc.** **T and J Software**

Support your Canadian dealers:

COMPUTER DOWNLOAD UNLIMITED
8 Talon St.
North Bay, Ontario
P1A 1N5
(705) 476-9391
Data (TEXLINK) 476-3043
CompuServe "73657,3617"
DELPHI I.D. "CDU"

DATA*PORT
2846 Gottingen St.
Halifax, N.S.
B3K 3E1
(902) 454-0232
Data (TEXLINK) 455-2076

Dealer enquiries welcome!

FROM
P.O. BOX 2144, STATION D, OTTAWA
*** ONTARIO, CANADA K1P 5W3 ***

EDMONTON 99er USER'S GROUP
P.O. BOX 11983
EDMONTON, ALBERTA
T5J 3L1